

**RL-TR-97-90**  
**Final Technical Report**  
**August 1997**



# **RESEARCH ON COOPERATIVE ACTIVE DATABASE SYSTEM**

**University of California, Los Angeles**

**Wesley W. Chu**

19971022 054

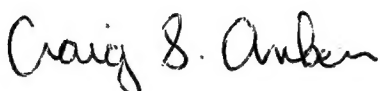
*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*


**Rome Laboratory**  
**Air Force Materiel Command**  
**Rome, New York**

**[DTIC QUALITY INSPECTED 5]**

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-97-90 has been reviewed and is approved for publication.

APPROVED:   
CRAIG S. ANKEN  
Project Engineer

FOR THE DIRECTOR:   
JOHN A. GRANIERO, Chief Scientist  
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL/C3CA, 525 Brooks Rd, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE Aug 97	3. REPORT TYPE AND DATES COVERED FINAL Sep 95 - Sep 96	
4. TITLE AND SUBTITLE  RESEARCH ON COOPERATIVE ACTIVE DATABASE SYSTEM		5. FUNDING NUMBERS  C - F30602-95-1-0052 PE - 62232N PR - R427 TA - 00 WU - P2	
6. AUTHOR(S)  Wesley W. Chu			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  University of California, Los Angeles Dept of Computer Science 405 Hilgard Ave. Los Angeles, CA 90095		8. PERFORMING ORGANIZATION REPORT NUMBER  A95-3061A-00	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Rome Laboratory/C3CA 525 Brooks Rd. Rome, NY 13441		10. SPONSORING/MONITORING AGENCY REPORT NUMBER  RL-TR-97-90	
11. SUPPLEMENTARY NOTES  Rome Laboratory Project Engineer: Craig S. Anken/C3CA/4833			
12a. DISTRIBUTION AVAILABILITY STATEMENT  Approved for public release; distribution unlimited.		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  Active database systems are receiving increasing interests from both research and commercial communities. However, rules are often difficult to specify and the complexity increases as the database increases in its size. To remedy this problem, we propose to use relaxation techniques for rule generation and relaxation. By using high-level concepts and cooperative operators in active rules, we can not only simplify the rule specification process, we can also increase the expressiveness of active rules. High-level concepts and cooperative operators used in rules are first relaxed into low-level active rules by using a tree-type knowledge structure called Type Abstraction Hierarchy (TAH). The relaxed rules are then classified into equivalent classes by domain experts. Rule generation and relaxation are accomplished by relaxing the attributes in the rule conditions and/or by relaxing the actions with cooperative operators. This report presents a design concept of Cooperative Active Database Systems and their future research directions.			
14. SUBJECT TERMS Cooperative rules, cooperative operators, rule relaxation, active database systems, cooperative active database systems		15. NUMBER OF PAGES 16	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT  UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE  UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT  UNCLASSIFIED	20. LIMITATION OF ABSTRACT  UL

# Cooperative Active Database Systems

## 1 Objective

Active database systems are receiving increasing interests from both research and commercial communities. However, rules are often difficult to specify and the complexity increases as the database increases in its size. To remedy this problem, we propose to use relaxation techniques for rule generation and relaxation. By using high-level concepts and cooperative operators in active rules, we can not only simplify the rule specification process, we can also increase the expressiveness of active rules. High-level concepts and cooperative operators used in rules are first relaxed into low-level active rules by using a tree-type knowledge structure called Type Abstraction Hierarchy (TAH). The relaxed rules are then classified into equivalent classes by domain experts. Rule generation and relaxation are accomplished by relaxing the attributes in the rule conditions and/or by relaxing the actions with cooperative operators.

## 2 Motivation

Incorporating rules into database systems has been a major focus in the database research community for more than a decade, the initial focus on deductive rules and integrity constraints has been augmented by recent research into active rules [WC95]. In active database systems [WC95, Kim95, BM91, Cha89, Day88], rules are defined and stored in databases. A rule in active databases usually includes three parts: an event, a condition and an action. When the triggering event occurs, if the condition is met, then the corresponding rule is fired and certain actions are executed. By incorporating rules into database systems, we can transform a passive database system into an active one. The database itself can respond automatically to internal or external events and take appropriate actions when those events occur. Because active databases can respond to events automatically, many tasks which either could not be done in traditional database systems, such as general integrity constraints maintenance and work-flow management, or have to be done by special subsystems in traditional systems, such as simple integrity constraints checking and view maintenance, can now be managed easily by active database systems. This is one of the reasons why many researchers have turned their research attentions to active database area.

For a large active database, the process of designing active rules are quite difficult and time consuming. What we would like to have is a facility through which a high level specification of rules can be automatically generated into low level rules. Furthermore, a lot of database domain knowledge is needed to clearly specify all the rules. For example, users have to know the details of the schema definition and information about data stored inside database. Very often user does not know how to exactly represent what they want, but they can specify their requirements with

regard to some other existing rules. For example, the cooperative operators “similar to”, “close to”, “near to”, and “approximate” can be included in users’ specifications. Finally, without any structure the low-level rules created by users themselves may be conflicting and are difficult for optimization. If rules can be generated from high-level specification, the system can guarantee that all the rules generated are consistent, and are easier for optimization.

Currently most research attentions have been focused in the area of event specification and detection, rule execution semantics and implementation methods of active databases. Very little attentions have been placed in the area of extending condition and action parts of the active rules. Most systems assume the condition part of active rules is just ordinary database predicates and action part is just database operations and user-defined applications. This proposal mainly focuses on extending the condition and action parts of active rules, by introducing cooperative operators and high-level concepts in these two parts. Furthermore, we also allow users to specify which active rules are relaxable so that when the conditions of these rules are not met, alternative rules can be applied.

We make use of a tree-type knowledge structure, Type Abstraction Hierarchy (TAH) [CCL90, CMB93], which is generated from database. The database knowledge represented in the TAHs is user and application contexts sensitive. Since the TAHs can be generated automatically from database, it is also scalable.

### 3 Rule Enhancements

Traditional active rules are precise and they directly interact with low-level database attributes. Hence, rule designers and application users have to have detailed knowledge about underlying database in order to specify active rules. However, this is usually not the case in the real world. Rule designers and application users either do not pertain such detailed knowledge of underlying database, or even if they have the knowledge, the task of specifying active rules using that knowledge is often difficult. What we propose here is to enhance active rules using relaxation techniques. By using relaxation techniques, rule designers and application users can use high-level, approximate rule constructs in their rule specifications, even though they might not have the detailed knowledge about the underlying database. The underlying database knowledge can be either automatically generated from database itself or supplied by domain experts.

#### 3.1 High Level and Fuzzy Rules

As human being, we tend to use high level and fuzzy concepts in our daily life. For example, instead of saying if the temperature is greater than 90F, we will go swimming, we say if the temperature is high, we will go swimming. Here ‘high’ is a high level and fuzzy concept. The exact meaning of high temperature depends on context. The current research on active database often ignores this fact. For example, consider the following rule:

R1: If `wind_speed > 20` and `wave_height > 5` then notify commanders.

R1 is a precise rule which uses exact number to represent the rule triggering condition. The drawback of this is as follows. 1) The rule designers have to have detailed knowledge as to what kind of weather condition is considered as a bad weather and hence need to notify commanders. 2) This rule ignores the fact that `wind_speed` is not a precise concept (we could not possibly measure the exact speed of wind). Hence, what about `wind_speed` of 19.9? Do we consider `wind_speed` of 19.9 as an indicator of bad weather? To remedy this, we propose to use fuzzy and high level concepts in active rules. For example, instead of R1 we could use:

R1': If the weather is bad, then notify commanders.

In this rule, the bad weather is a high-level and fuzzy concept. Based on different user profile and application context, we could build different knowledge structures of bad weather (in weather TAHs). For example, for commanders involved in a plan segment, we could define bad weather as high wind speed and high wave height (see Figure 1). For grade students, we define bad weather as either high temperature, low temperature, snow, rain, wind, or any combinations of them. In this way, the rule designer does not need to have the detailed knowledge about the underlying database because all these knowledge can be either automatically generated from database, or supplied by domain experts.

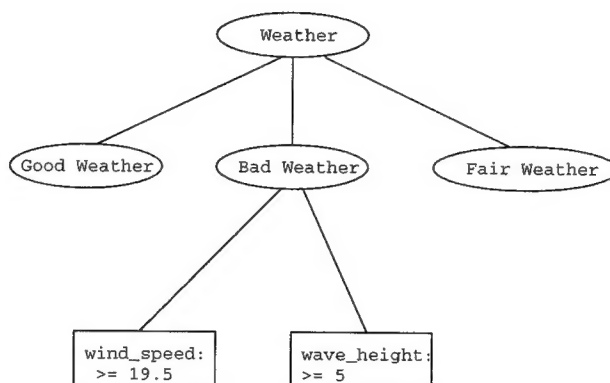


Figure 1: TAH for Weather Condition for R11

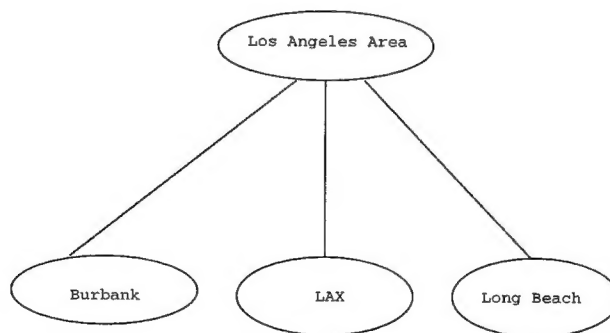


Figure 2: TAH for Los Angeles Area for R2

By using high level active rules, we also limited the number of rules in the system. Instead of specifying all low-level rules at the user level, the system can automatically generate low-level rules from those high level rules supplied by users. For example, the following rule:

**R2: Notify user if A flight from Los Angeles area to New York is inserted into database.**

can be rewritten into the following rules according to our knowledge about Los Angeles area (see Figure 2):

**R21: Notify user if there is flight from LAX to New York is inserted into database.**

**R22: Notify user if there is flight from Burbank to New York is**

inserted into database.

R23: Notify user if there is flight from Long Beach to New York is  
inserted into database.

### 3.2 Rules with Cooperative Operators

In active databases, all rules are expressed in a precise and exact fashion. Often there is a need to express rules in more general terms. Cooperative operators such as *approximate*, *similar-to* and *near-to* [CCL90] can be used in the rules to enrich the expressive power. Consider the following examples,

R3: If the weather turns bad, notify all affected units in that region.

This rule says that if we have a severe weather in a region, we need to notify all units currently in that region. However, this rule does not consider that fact that all neighboring units to that region might also be affected by this bad weather. If we allow cooperative operators in active rules, we could have,

R3': If the weather turns bad, notify all affected units in that region and all those units that are *near-to* that region.

This rule uses cooperative operators *near-to* in the action part to extend the action. Consider another example,

R4: If find an airport *similar-to* Bizert airport based on runway length and runway width, then notify commanders.

This rule uses the *similar-to* operator to incorporate a fuzzy concept.

From the above examples, we note that the expressiveness of our active rules is greatly enhanced by introducing cooperative operators into active rules.

### 3.3 Rules with Relaxable Conditions

For traditional rules, if the condition does not hold, the rule will not be fired. However, by relaxing the condition part of the rule, more intelligent behaviors can be provided. Consider the following rule:

R5: If the there are 100 combat aircrafts in region 1, then notify commanders.

If we use ordinary active rule to represent R5, the commanders will not be notified until there are 100 combat aircrafts in that region. However, by using database knowledge (TAHs) on the geographical database of region 1, we could automatically relax the condition of this rule to the following,

R5': If there are 100 combat aircrafts in region 1 and its *nearby* regions, then notify commanders.

There are two approaches to rule relaxation. One is explicit relaxation while specifying the relaxation condition in the rules. Another approach is implicit relaxation in which the system will

provide such relaxation automatically even if the relaxation condition is not explicitly specified in the rules. The explanation system will then inform the users (e.g., commanders) of such relaxation.

## 4 Processing Enhanced Rules Using Relaxation Technique

To enhance an active database with cooperative operators and high level concept, we propose the following rule processing architecture (Figure 3).

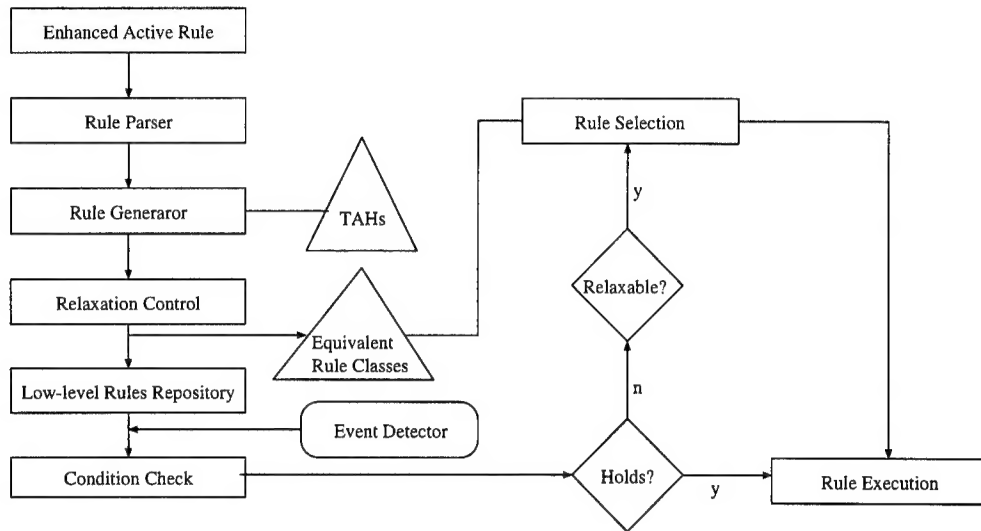


Figure 3: Processing enhanced rules with relaxation techniques

The enhanced active rules are parsed by the *Rule Parser* which recognizes the high-level concepts and cooperative operators used in these rules. A *Rule Generator* will then apply the knowledge from database (via TAHs) and generate ordinary active rules from those high level and approximate rules. If the rule is relaxable (can be specified by users when defining these rules), the conditions of the rule is then relaxed by using attribute TAHs from its condition part. The generated rules will then go through the process of *Relaxation Control*. This relaxation control process classifies rules into equivalent classes so that during rule processing an equivalent rule can be applied if the condition part of the original rule does not hold. The purpose of this relaxation control is to put a control on the rule generation/relaxation process. After the relaxation control process, the low-level rules are stored in a rule repository.

Now when an event is detected by the *Event Detector*, the rules will be chosen for condition evaluation. If the condition holds, the ordinary rule execution will process. Otherwise, a relaxed rule is selected from the equivalent rule class and the relaxed rule is then sent back for rule execution.

## 5 Rule Generator

The *Rule Parser* recognizes rules with cooperative operators such as *similar-to*, *near-to*, *approximate*, and high level concepts. The parsed result is sent to *Rule generator* which uses the Type Abstraction Hierarchy (TAH) to guide its rule generation. Rule generation can be done on condition part based on the attributes in the condition, using the corresponding TAH. A set of low-level rules may be generated in this way. Rules can also be generated on action part of the rule by



using cooperative operators such as *near-to*, *similar-to* and *approximate*, and thus generate a set of low-level rules. The TAHs can be either generated automatically from database attributes or supplied by domain experts. Note that the approximate operators and high level concepts can appear both in condition part and action part of an active rule.

## 6 Rule Relaxation and Control

Similar to query relaxation, rule relaxation process also uses the TAHs from database when relaxing condition part of an active rule. The relaxation process is only done when user explicitly indicates this rule is relaxable (parser needs to recognize such indication). In this way, we only relax those rules which are specified relaxable by users and hence avoid the unexpected consequences resulted from excessive rule relaxation. During the relaxation process, an alternative rule is chosen from the equivalent rule set to be executed. The set of rules will have relaxed condition/action part, but will be considered as equivalent in that application context.

### 6.1 Equivalent Rule Classes

Rule processing in active databases is different from query processing in that rule processing does not need user's interactions during the execution of the active rules. Control must be provided for the rule generation/relaxation process, since the uncontrolled relaxation may produce unexpected result.

An ECA rule consists an event, a condition and an action. When we consider rule generation/rule relaxation, we need also to consider which part of the rule to relax first and what impact the relaxation on one part (e.g., condition) does on the relaxation on another part (e.g., action). For example, suppose we have the following rule:

Event: On a  
Condition: if b  
Action: c

and condition b is semantically equivalent to b1, action c is semantically equivalent to c1, then is the rule

Event: On a  
Condition: if b1  
Action: c1

still equivalent to the original rule?

In order to solve this problem, we propose to add relaxation controls into our rule generation/relaxation. First, after the rule generation process using the TAHs, an algorithm will be used to classify rules into equivalent rule sets. We classify two rules as *semantically equivalent* if:

- the attributes in the condition/action parts contain similar concepts as defined by their respective TAHs.
- they are the result of generation of a high level rule/approximate rule with cooperative operators.

After all equivalent rule sets have been generated, the domain experts/rule designer can classify them into equivalent classes in the sense that all rules in a set can be relaxed between one and another and generate equivalent (semantically equivalent) effects. The equivalent effects can also be classified into different levels such as strongly, moderately and weakly equivalent, etc.

## 6.2 User Profile and Application Context

Similar to query relaxation in CoBase [CMB93], user profile and application context play an important role in the process of rule generation and relaxation. Different users or applications generate/relax rules differently. For example, for the high level concept **bad weather**, one user might mean high wind speed and high wave height, while another might mean rainy days. Hence, TAHs must be generated according to user profiles and application contexts. Furthermore, users can also specify relaxation control to indicate which part of the rule is relaxable or non-relaxable to satisfy the user's requirements and to reduce ambiguities.

## 7 Future Research Directions

- Rule Generation
  1. Develop methodology of generating basic rules from rules with high level concepts in the condition and action parts
  2. Develop techniques to generate basic rules during compile time
- Rule Relaxation and Control
  1. Incorporate relaxation operator into CONDITION part of the rule
  2. Incorporate relaxation operator into ACTION part of the rule
  3. Incorporate relaxation operator into EVENT part of the rule
- Relaxation Control
  1. Provide a rule language specifications with cooperative and relaxation control operators
  2. Develop a technique to classify the generated rules into equivalent rule classes
  3. Develop relaxation control via different levels of equivalent classes
- Implementation Plan
  1. Generate a plan to implement a prototype with relaxation features into active database system (e.g., OODB and Sentinel) to study the behavior of such enhanced active system.

## References

- [BM91] C. Beeri and T. Milo. A Model for Active Object Oriented Database. In *Proceedings of the 17th International Conference on Very Large Data Bases*, Barcelona, September 1991.
- [CCL90] Wesley W. Chu, Qiming Chen, and Rei chi Lee. Cooperative Query Answering via Type Abstraction Hierarchy. In *Proceedings of the International Working Conference on Cooperative Knowledge Based Systems*, October 1990.
- [Cha89] S. Chakravarthy. Rule Management and Evaluation: an Active DBMS Perspective. *SIGMOD RECORD*, 18(3), 1989.
- [CMB93] Wesley W. Chu, M. A. Merzbacher, and L. Berkovich. The design and implementation of cobase. In *Proceedings of ACM SIGMOD*, Washington D.C., May 1993.

- [Day88] U. Dayal. Active Database Management Systems. In *Proceedings of the 3rd International Conference on Data and Knowledge Bases*, Jerusalem, Israel, June 1988.
- [Kim95] Won Kim, editor. *Modern Database Systems*, chapter 21. Addison-Wesley Publishing Company, Inc., 1995.
- [WC95] Jennifer Widom and Stefano Ceri. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann, 1995.

## ***MISSION OF ROME LABORATORY***

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Material Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.